Attorney Docket No. 16869B-064300
Client Ref. No. HAL-ID 239

PATENT APPLICATION

Database Query Operations Using Storage Networks

Inventors: **Yuichi Yagawa**
Citizenship: Japan

Assignee: **Hitachi, Ltd.**
6, Kanda Surugadai 4-chome
Chiyoda-ku, Tokyo, Japan
Incorporation: Japan

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 650-326-2400

Database Query Operations Using Storage Networks

BACKGROUND OF THE INVENTION

[0001] This invention relates to methods and apparatus for querying databases and obtaining the results from such queries. In a typical database management system, an application system is used to generate queries and present results retrieved in response to those queries from a database system. Numerous vendors such as Oracle and IBM, provide relational database management systems. Many vendors also provide application systems or application servers, for example, implementing them as a Java Servlet. An example of a query is a request made in structured query language or SQL.

[0002] Figure 1 is a block diagram of a typical current state of the art system in which an application system queries a database system. As shown there, the application system 10b is connected to a database system 40b over a network 1b. Network 1b will typically be provided by a local area network (LAN) or a wide area network (WAN). Both the application system in 10b and the database system 40b are connected to a storage system 80b. Although in Figure 1 the storage system is depicted as being coupled to both the application system and the database system, in many installations separate storage systems will be provided for each of the application system and the database system. Another network 2b is dedicated to data transfer between the application and database system and the storage system. The communication protocol for the storage system will often be Fibre Channel (FC), network file system (NFS) or other file systems such as iSCSI or other IP-storage protocols. While figure 1 illustrates the system as encompassing only a single application system 10b and database system 40b, in many conventional implementations there are configurations in which several application system and several database systems are interconnected.

[0003] In Figure 1 the primary components of the application system include a query requestor 20b that, in response typically to a user request, creates and sends a query to the database system 40b through the local area network 1b. The application system also typically includes some means for receiving the results 30b, typically through the LAN connection 35b. A typical prior art system the database system 40b includes a query receiving means 50b for receiving queries from the application system 20b through the LAN. It also includes query executing means 60b to analyze the content of the query and execute the appropriate

processes. Once the answer to the query has been determined, a process for returning the results 70b returns the result of the query to the application system 10b so that the answer to the query may be presented to the user.

[0004] Application and database systems such as depicted in Figure 1 are capable of handling enormous amounts of data, for example, as might be found in a bank or an airline reservation system. This enormous amount of data is stored in a storage system 80b which includes database volumes 81b, which manages the database files 82b. The database files contain the actual data, such as the ticket numbers of all the passengers on a given flight.

[0005] The system such as depicted in Figure 1 operates by having a query 5b sent from the application system to the database system through the local area network, or other network. As shown by arrow 7b, the database system then interacts with the storage system to retrieve the desired information, and return it as shown by arrow 6b over the local area network back to the application system. In Figure 1 the most significant processes with regard to the invention described below are shown in the darker rectangles. These processes include sending the query 25b (or the query in the form of data), receiving the query (or the query in the form of data) 55b, returning the result data 75b, and receiving the result data 35b.

[0006] A substantial disadvantage of the system depicted in Figure 1 is the requirement that the data flow back and forth over the local area network. When the result from the database system to be provided to the application system contains a large quantity of data, the local area network becomes a bottleneck for overall performance. For example, if the query is for a list of all of the passengers who have traveled on a particular airline in the last ten years, a very large amount of data will be retrieved. In these situations, the large amount of data required to traverse the network limits the operating speed of the overall database system. In addition, although less frequently, sometimes the query itself contains a large amount of data and this also creates a bottleneck.

[0007] Accordingly, what is needed is a system for implementing queries of databases which avoids the bottlenecks introduced by sending the query over a local area network, or retrieving large amounts of data in response to the query over the network.

BRIEF SUMMARY OF THE INVENTION

[0008] This invention provides a system in which query data and query results, rather than being transferred between a database and a application system are mutually accessed by being stored in a shared volume of a storage network. A key is used to specify the location for the stored data and enable access to it by the intended user of the data. The system

minimizes use of the LAN, or other type of local, wide-area, internet, or other network connection, for the transfer of large amounts of data, thereby improving operation of large database systems.

[0009] In one embodiment a system for enabling queries to a database to be processed includes an application system for providing queries to a database system coupled to the application system via a network, a storage system coupled to each of the application system and the database system, and a return path selector coupled to the database system for selecting a return path over which to return results from queries made to the database system. The return path selector selects between the network connection and the shared storage system, usually based on the size of the result file to be transferred. The storage system may be coupled to the application system and the database system using a switch or a storage network, and a database hub may be used to couple the application system and the database system.

[0010] In another embodiment the application system includes a request path selector for selecting a request path over which to send query data for requests made to the database system. The request path selector selects between the network connection or the storage system, usually based on the size of the query data file. If the storage system is selected the query data is stored there and a key sent over the network to the database system to enable access to the query data. In legacy systems a database gateway may be used in the database system, and a database access program in the application system to provide the functionality of the foregoing two embodiments. As with the preceding embodiment, the storage system may be coupled to the application system and the database system using a switch or a storage network.

[0011] In another embodiment, in a system having a query provider which provides queries to a database system over a network, the query provider and the database system being coupled to a storage system, a method of returning results to the query provider includes storing the results in the storage system and sending the address of the results over the network to the query provider. In some implementations of the system the at least one of the key and the results are encrypted. In other embodiments the query data may be stored in the storage system, either through a storage network or a switch, and its location transferred over the network to the database.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Figure 1 is a block diagram illustrating a prior art system architecture for an application system and database system;

[0013] Figure 2 is a block diagram of an embodiment in which query results are returned using a storage network;

[0014] Figure 3 is a block diagram of an embodiment in which query data is provided using a storage network;

[0015] Figure 4 is a block diagram of an embodiment in which database access and gateway software is employed to provide query results;

[0016] Figure 5 is a block diagram of an embodiment in which database access and gateway software is employed to provide query data;

[0017] Figure 6 is a block diagram of an embodiment in which a switch is used to couple the database system, application system and the storage network;

[0018] Figure 7 is a block diagram of the system architecture of an embodiment in which a database hub system is employed;

[0019] Figure 8 is a flow diagram of a process for selecting return paths;

[0020] Figure 9 is a flow diagram of a process for storing result data;

[0021] Figure 10 is a flow diagram of a process for accessing result data;

[0022] Figure 11 is a flow diagram of a process for selecting a request path;

[0023] Figure 12 is a flow diagram of a process for storing query data; and

[0024] Figure 13 is a flow diagram of a process for accessing query data.

DETAILED DESCRIPTION OF THE INVENTION

[0025] This invention provides a system in which query data, rather than being sent to a database via a network or returned via a network connection between the application server and the database system, is stored in a shared volume of a storage network. The address for that stored data is then used to access it by the application system and/or the database system. In this manner the application system is permitted to access the address on the shared volume and receive through the storage network the query data and/or the result data associated with an original query.

[0026] Figure 2 illustrates a first embodiment of the invention. In this embodiment the response time for returning large amounts of data in response to a query applied to the database system by the application system is greatly reduced. While the overall architecture

shown in Figure 2 at first appears similar to that of the prior art, some of the processes and the operations are different from the system depicted in Figure 1.

[0027] Storage system 80a includes shared volumes 90a which are shared between the application system 10a and the database system 40a. Note that the shared volume is typically, but not necessarily distinct from the database volume 81a in which the database file 82a is stored. Shared volume 90a is shared between the application system 10a and the database system 40a using well known storage system technology. The shared volume is capable of storing the result data file 93a which is the data file created by the application of a query to the database system. That result data file has associated with it a key field 91a and a flag 92a as shown in Figure 2.

[0028] In a typical embodiment of this invention employing shared volumes, it will be necessary to mount and un-mount the volumes depending upon their use. For example, before the application system 10a accesses the shared volume 90a, the volume needs to be mounted. Before the database system accesses the shared volume 90a, the application system must un-mount the volume 92a, and the database system must mount the volume 92a. These are conventional actions in a storage system and are not discussed further herein.

[0029] The manner of operation of the system shown in Figure 2 is discussed next. In a well known manner, typically with input from a user addressing the application system from a terminal, a query is formulated to be applied to the database system. In general the purpose of the query will be to retrieve information from the database which is related to the particular interest of the user. For example, if the database system is storing information about product sales of a particular corporation, a query might seek all of the products sold under a particular model number between January and May of a particular year by retail outlets situated in a particular telephone area code. The query 5a, after being formulated by the user, is transmitted over a network 1a, typically a local area network or LAN, to the database system 40a. The query is received by a query receiver 50a which may buffer the query depending upon the workload of the database system at the moment, or it may otherwise pre-process the query.

[0030] Once the query is ready for execution, the query is supplied to the execution unit 60a which itself includes a process for selecting a return path 61a for the query results. This specific process employed is described below. In general, however, the process for returning results 70a will cause the results to be returned through the local area network using process 72a, or will cause them to be returned by providing data to the storage system as shown by process 71a. As will be discussed in more detail below, selection of the particular

path for returning the result data will typically involve consideration of the amount of data to be returned. In the example given above, if only one, or a few records, are to be returned to the application system, they will typically, or preferably, be returned over the local area network 1a. On the other hand, if the results are voluminous, they may be returned by being stored into the shared volume 90a where they are accessible by the application system. After the results have been returned, they are received by the application server by process 30a. If the results are to be returned over the LAN, for example by process 75b in Figure 1, they are handled in the conventional manner (shown in Figure 1 by process 35b). On the other hand, if the results have been returned to the shared volume 90a then the result receiver 30a in the application system will receive the address of the result data file 93a from the database system 40a. The result receiver 30a may then use the address received to retrieve the result data from the shared volume 90a.

[0031] The particular processes for operation of the architecture of Figure 2 are shown by dotted lines. The request of the query from the application system 10a to the database system 40a is illustrated by process 5a. Process 7a illustrates the access of the data from the database volume by the database system using the storage system. Process 8a shows the results of the query being stored as a result data file 93a in the shared volume 90a by use of the storage network. Process 6a illustrates the return of the address of the result data file 93 from the database system 40a to the application system 10a through the LAN. Using the address received over the LAN, process 9a illustrates the access of the result data in the storage by the application system.

[0032] As mentioned, the shared volume 90a is an aspect of the system which enables avoiding transmitting the result data through the LAN. This shared volume is formatted before the application system and the database system may access it. Typically, the size of the shared volume 90a is defined at the time it is formatted. If desired, an optional feature of the invention allows that for each result data file 93a the shared volume to maintain a key (or lock) 91a which may be unlocked by an appropriate code sent to it, as well as a flag 92a.

[0033] The code which unlocks key or lock 91a usually will be unique for each access to the database. It is exchanged between the service requesting a process in the application system and the service providing the process in the database system 40a. The key 91a helps to protect other processes from accessing the result data file 93a. Preferably the key 91a will be stored in a special area of the shared volume which is accessible using the address of the result data file 93a. Of course, if additional security is necessary or desirable, the result data itself may be encrypted in the file using the key or other desired technique.

[0034] The flag 92a records each state of the result data 93a so that the application system and the database system can access the data exclusively and appropriately without conflicting with each other. The states for the flag are Writing, Ready, Reading, and Done. The Write state is the one in which data is written to the file. During this time the application system is not permitted to access the data. Ready indicates that the data has been completely written and may now be read. Read is the state in which the data is read from the result data file by the application system. Done is the state in which all of the data has been read from the file and the process is complete. Flag 92a is stored in the special area of the shared volume 90a which is accessible using the address from the result data file 93a, or the flag may be stored in a special area of the result data file 93a itself. Preferably the shared volume will be large enough to accommodate many result files at the same time, however, this will depend upon the available density of hard disk drives and the quantity of results stored.

[0035] Next we discuss the approaches for implementing some of the steps shown in the diagram of Figure 2. The manner in which the process of selecting a return path 61a is implemented is shown in more detail in Figure 8. As shown there, the first step after starting the process is to retrieve the result data. The result data is obtained by applying the query received from the query receiver 50a to the database file 82a using well known query execution techniques 60a. Once the result data is obtained, the next step is to calculate the size of the result data. After this calculation, the size of the result data is compared against the threshold value at step 202. If the size is greater than the threshold value, then step 203 will be selected. Otherwise, step 204 will be selected. Step 203 chooses a return path for providing the data to the shared volume as shown by process 8a in Figure 2. Step 204 sets the return path to be the conventional one, which is back to the application system through the LAN. This is shown by process 75b in Figure 1.

[0036] The threshold value for use in step 202 of Figure 8 may be predetermined, or may be determined at the time on the basis of the current workload of the LAN or other limiting parameter. In alternative implementations the processing of setting the return path may not wait for the calculation of the size of the return data as shown by step 201. Instead, the process may make an assumption of the size of the return data based upon similar queries from the query execution history.

[0037] In Figure 2 the process for sending the result data to the storage 71a is shown as a part of the return results process 70a. Figure 9 is a more detailed diagram illustrating the process for sending the result data to the shared volume for storage. In an initial step 220 the size of the result data is calculated. Of course, if that result data size is already calculated by

process 61a in Figure 2, it is not necessary to recalculate it in step 220. Once the size of the result data is known, then in step 221 a suitable file space is allocated in shared volume 90a for the result data file 93a. As shown by step 222, if there is enough space in the shared volume, then the process flow may continue. If there is not enough space in the shared volume, the process shifts to step 223 in which "garbage collection" in the shared volume is performed to clean up the shared volume and provide additional storage space sufficient to hold the file. As mentioned earlier, if the flag for result data files previously stored on the shared volume has been set to "Done" then those files may be deleted from the shared volume to make additional space. The files may be deleted in any specified order, for example, oldest to newest, biggest to smallest, user set priority, etc.

[0038] Once enough space is available, as shown by step 224, the flag is set to Write for the particular file. This precludes access to the file by any other process while it is being written from the database system to the shared volume. After setting the flag, the data itself is stored in the shared volume. If desired, the data may be encrypted.

[0039] Once the data is stored a key is established for that file as shown by step 226. The key is unique among all of the result data files and may be created using any well known algorithm. For example, in a preferred embodiment a hash algorithm for the query message itself is used to create a key. After all the data is written to the file, the "Ready" flag is set for that file, as shown by step 227. The setting of the Ready flag enables further processes in the application system to access the result data. Finally, the process is complete when the address of the result data file 93a and the key 91a are sent back to the application system by the database system. This is shown by process 6a in Figure 2. The process 6a of returning the address and key may be carried out in a conventional manner, except of course that the accompanying data is not transferred at the same time. If additional security is necessary, the return address and key may themselves be encrypted for transmission.

[0040] Continuing with respect to Figure 2, process 31a receives the address (and the key) through the LAN. The result data stored in the shared volume may then be accessed. Figure 10 is a diagram illustrating the process for accessing the result data file 93a in the shared volume 90a. This process corresponds to process 9a and process 32a in Figure 2.

[0041] As shown in Figure 10, the first step 240 is to obtain the storage address and the key for the result database file from the database system through the LAN. The application system then checks to see if the key received from the LAN matches a key 91a stored in the shared volume for the different result data files 93a. Once the key is found which matches the received key, as shown by step 242, the process continues with checking

the flag 92a, as shown by step 244. If the key does not match, then an error is returned. The flag is accessed using the address received. If the flag is other than "Ready" the process continues checking the flag until it becomes "Ready". This mechanism allows both the database system and the application system to exclusively access the result data file, thereby guaranteeing data integrity. To prevent an infinite loop, a counter may be employed to determine the number of times the flag is rechecked. The system then returns an error if the number of times exceeds a predefined threshold.

[0042] As shown by step 246, once the flag is set to Ready, the application system resets it to "Read" thereby precluding access to the data file by any other process. At step 247 the file is read, and decrypted if necessary. After reading, as shown by step 248, the flag is reset to "Done" to indicate that all of the data has been read from the file. As discussed earlier, the setting of the "Done" flag will allow the garbage collection step 223 to delete the file when space is required on the shared volume.

[0043] It is sometimes the situation in the use of relational databases that the user will want to re-query the result data. For example, the user may wish to narrow the results by applying an additional query, in effect adding one further limitation to the previous query. In the hypothetical situation discussed above, for example, the user may wish to narrow the query to only request sales made during the evening. In situations like this it is advantageous that the result data file 93a be accessible from the database system 40a to facilitate re-querying the data without need to re-execute the original query. Such a capability improves the response time to the user.

[0044] This re-querying is enabled by having the application systems send back the key in the address of the result data file 93a with a subsequent re-query. When the database system receives the re-query request, the database system checks if the result data file 93a for that particular query still exists in the shared volume 90a. If it does not exist, then the database system must re-execute the first query and either simultaneously, or as a subsequent step, execute the additional query. (The determination of the precise manner of execution of queries is defined by the database system software provider and is not discussed here.) The system then checks to assure that the appropriate key is present and that the flag is set to "Done." As before, the system will wait only a certain amount of time for the flag to be set to "Done." If the wait time exceeds the pre-determined threshold, then the system will execute the first query and then the re-query. Assuming the operation is as expected, however, the system will execute the re-query using the result data file. Then, using the above-described

techniques for storing the result data file back into the shared volume, a new result data file will be stored with a new key on the shared volume for access by the application system.

[0045] Figure 3 is a block diagram illustrating another embodiment of the system of this invention. This embodiment improves the response time when large amounts of query data need to be sent from the application system 10c to the database system 40c. As shown in Figure 3, the overall system architecture is very similar to that depicted in Figure 2, but with some of the processes being modified. One modification is that the query requestor 20c includes a process 21c for selecting the request path and sending the query data to the database system 40c. The query requestor includes a process for storing the query data 22c and sending the address for the query through the LAN 23c.

[0046] In the database system the query receiver 50c includes a process for receiving the address with the query through the LAN 51c. In addition, the query execution process 60c includes a process 62c for accessing the query data stored in the shared volume 90c. As with Figure 2, the dotted lines in Figure 3 indicate various processes. Process 9c represents storing of the query data 22c as a query data file 93c in the shared volume of the storage system 80c. As this occurs, the application system 10c obtains the address of the query data file 93c. Process 5c represents a request of the query from the application system to the database system through the LAN. The query will contain the address of the query data file 93c. Process flow 8c represents access of the query data file using the address received from the LAN. Process 7c represents sending the query data to the database volume 81c, while process flow 6c represents a return of the result of the database process from the database system through the LAN.

[0047] Specific components of Figure 3 are discussed next. Other than the query data file, the shared volume will have the same characteristics as in the first embodiment of Figure 2. The query data file 93c allows the exchange of query data from the application system to the database system. This is particularly useful for queries where a large amount of information is required for the query. For example, a list of thousands of telephone numbers and a single address may be the query, with the desired result being a determination of whether any of the people having those telephone numbers live at the specified address. The name of that one person can easily be returned over the LAN. In addition, the storage network can be used for circumstances in which both the data for the query and the result from the query contain large amounts of data. For example, in a situation where a list of thousands of employee identification numbers are in the query and the request is to obtain the office telephone number for each employee. Such an implementation combines the ideas

depicted in Figures 3 and , and is a situation where both the query and the result use the storage network.

[0048] Another difference between the embodiment of Figure 3 and that of Figure 2 is the select request path process 21c within query requestor 20c. The select request path process is described in more detail with respect to Figure 11. As shown in Figure 11, the process begins with collection of the query data at step 300. The size of the query data is then calculated at step 301. At step 302 the size is compared with the threshold of value at step 302. If the data file is smaller than the threshold value, then the request is sent over the LAN to the database system as shown by step 304. On the other hand, if the size is large, then the request path will be set to be the storage network. As before, the threshold for this choice may be set as an absolute number, or based on relative availability of various system resources.

[0049] Figure 12 is a flow chart of process 22c in Figure 3 in which the query data is stored in the shared volume. The process begins with the step of obtaining the size of the query data 320. If this has previously been determined in conjunction with the selection of the request path, then it is not necessary to be repeated. If it is not, then it is computed. As shown in step 321, once the size is determined, a suitable amount of file space in the shared volume 90c is allocated. This is done by determining whether there is already enough space in the shared volume or not. As shown by step 322, if there is not, the garbage collection step 323, similar to step 223 in Figure 9, is performed to make enough space. Once it is determined that enough space is available, the process sets the Write flag at step 324 to preclude access by any other process. The query data is then stored to the shared volume as shown by step 325. The data may be encrypted if desired. In step 326 a key is assigned to the file, which, like the key for the other data stored in the shared volume, is unique. As also described, a hash algorithm may be used to compute the key. After the data is written to the file the flag associated with that file is reset to "Ready" as shown by step 327. The address and the key are then returned at step 328.

[0050] Returning to Figure 3, the query requestor 20c includes a process 23c for sending the address with the query through the LAN. This communication mechanism which transfers a request message from the application system to the database system is the same as discussed above with respect to Figure 2 here. However, the request message only contains the address and the key of the query data file 93c, instead of the entire data query itself. As mentioned above, in addition, if more security is required on the network, the address and the key may be encrypted before they are transferred. The database system query receiver 50c

includes a process 51c which receives the address with the query through the LAN. This is the same process as described above.

[0051] Once this information has been received by the database system, the next step is to access the query data which is stored in the shared volume 90c. This is process 62c shown in block form in Figure 3, and the details of its implementation are illustrated in Figure 13. The process begins at step 340 with obtaining the storage address and the key from the application. The database system then checks to determine if there is a match for the key it received, as shown by step 341. Step 342 illustrates that if the key does not match, an error is returned. If the key does match, however, the flag is checked at step 344. This flag is accessed using the received address from the LAN. As discussed above, if the flag is not "Ready" the process continues checking the flag until it becomes ready (subject to a time out as discussed above). This mechanism allows both the application system and the database system to exclusively access the query data file, thereby guaranteeing data integrity.

[0052] As shown by step 345, once the flag is detected as "Ready" it is reset to be "Read" thereby locking out other processes from changing the data. The file is then read, being decrypted if necessary, at step 347, and then the flag is reset to "Done." As also discussed above, resetting the flag to "Done" enables the garbage collection step 323 to delete this file when additional space on the shared volume is required.

[0053] It will be apparent that the first embodiment and the second embodiment discussed in Figures 2 and 3 above may co-exist in the same system. In such a circumstance, it will mean that each of the application system and database system includes all the components described in conjunction with Figures 2 and 3. It will also mean that the shared volume contains both result data files and query data files.

[0054] Figure 4 depicts another embodiment of a system of this invention. The embodiment in Figure 4 provides improved response time when large amounts of result data are requested from a request from a legacy application system made to a legacy database system. As shown by Figure 4, the implementation is similar to that discussed in conjunction with Figures 2 and 3, however, there are some additional aspects as discussed below.

[0055] The application program 100d is the legacy program, and usually will contain a process 101d for creating a query and a process 102d for processing results. The database access program 110d residing in the application system, allows the application system 10d to have the full features of the system of this invention without need for introduction of a new application program. The application program 100d accesses the database system 40d through the database access program 110d. Examples of database access programs 110d are

Open Database Connectivity (ODBC) drivers, Java Database Connectivity (JDBC) drivers, etc.

[0056] In this implementation the database access program 110d will typically include a process 111d for sending a query to the database through the LAN. It will also include a process 31d for receiving the address of the result data file from the database system through the LAN. This process will be the same process as process 31a described above. Database access program 110d will also typically contain a process 32d for accessing a result data file in the shared volume through the storage network. Preferably this process would be the same as process 32a also described above.

[0057] In this embodiment the database program 120d also preferably will be a legacy program. Usually it contains a process 121d for receiving a query from the application system, a process 122d for executing that query, and a final process 123d for returning the results of that query.

[0058] In this implementation a gateway program 130d is added to the database program 120d to allow the database system to have the features of this invention without need of introduction of a new database program. The gateway program 130d preferably includes a process 131d for "hooking" accesses to the database program 120d. This process 131d hooks every input/output transaction between the database program 120d and the application program 100d via the database access program 110d. The gateway also preferably contains a process 61d for selecting a return path for return of the result data. This process is preferably the same as process 61a. There will also be a process 71b for storing the result data to the shared volume through the storage network. This process is preferably the same as process 71a discussed above. Finally, the gateway program 130d will preferably also include a process 72d for returning the address and the key of the result data file from the gateway program 130d through the storage network. This process corresponds to process 72a discussed above.

[0059] The process flow for the embodiment depicted in Figure 4 is shown by the dotted lines in Figure 4. The process begins with a query request 140d from the application program 110d to the database program 120d via the database access program and the gateway program. Process 142d represents an access of the data in the database volume from the database program 120d. Process 143d illustrates the result data being passed from the database program 120d to the gateway program 130d. Process 144d illustrates storage of the result data file 93d in the shared volume from the gateway program. At the same time, the gateway program receives the address of the result data file 93d. Process 141d indicates the

return of the address of the result data file 93d from the gateway program to the database access program 110b via the LAN. Process 145d illustrates access of the result data file using the address received from the LAN, while process 146d illustrates passing result data from the database access program 110d to the application program 100d, completing the query.

[0060] Figure 5 is a block diagram of another embodiment of the invention. This embodiment improves the response time for sending extensive amounts of query data from an existing application system to an existing database system. As evident from Figure 5, the overall system architecture is similar to the system discussed in the second and third embodiments (Figures 3 and 4) but there are some additional features as discussed below.

[0061] As shown by Figure 5 the database access program 110e is added to the application program 100e to allow the application system to have full benefit of the ideas implemented by this invention without need for introducing a completely new application program. The application program 100e accesses database system through the database access program 110e. Examples of sample database access programs 110e are the ODBC and JDBC systems discussed above. The database access program 110e includes a process 21e for selecting a request path for sending query data to the database through the LAN. Process 21e typically will be the same as process 21c discussed above. It also includes a process 22e for storing query data to the shared volume in situations where the storage network is selected as the request path by process 21e. Process 22e will be the same as process 22c. Finally, the access program will include a process 23e to send both the address and the key of the query data file with the query itself to the database system through the LAN. Process 23e preferably will be the same as process 23c.

[0062] On the database side of the architecture, a gateway program 130e is added to allow the database system to have the full features of this invention without need for introduction of a new database system. Gateway program 130e preferably contains a process 131e for hooking accesses to the database program 120e. Process 131e will hook every input/output transaction between the database program and the application program. Process 51e receives the addresses and the key of the query data file with the query itself from the database access program 110e. Process 151e is preferably the same as process 51c. Process 131e is preferably the same as process 131d. Finally, the gateway program will also include a process 62e for accessing the query data in the shared volume using the address and the key. Process 62e is preferably the same as process 62c.

[0063] The dotted lines in Figure 5 illustrate the overall process flow of the embodiment. The process flow begins with passing the query data 146e from the application program 100e to the database access program 110e. Process 145e illustrates the method by which the storing of the query data as a query data file from the database access program 110e to the shared volume is performed. At the same time the database access program 110e will receive the address of the result data file 93e. Next, step 140e illustrates that the query is supplied from the application system to the database system through the LAN. The query will contain the address of the query data file 93e. Process 144e illustrates access of the query data file using the address received over the LAN. Process 143e illustrates the passing of the result data from the gateway program 130e to the database program 120e. Next step 142e illustrates transfer of the query data to the database volume 81e, and finally step 141e illustrates a return of the result of the database process from the database system to the application system through the LAN.

[0064] Additionally, as discussed above with respect to the first and second embodiments (Figures 2 and 3), the third and fourth embodiments (Figures 4 and 5) may co-exist in the same system.

[0065] Figure 6 illustrates another embodiment of the invention. This fifth embodiment employs a storage area network with a slightly different configuration than the embodiments discussed above. In particular, the fifth embodiment is similar to the first embodiment (Figure 2) but a switch, such as a Fibre Channel, is used to interconnect the application system and the database system with the storage systems. A characteristic of this architecture is that several server based systems, such as the application system and the database system depicted in the figure may share several storage systems. Thus the shared volume 90f may be present in a different storage system than the storage system 80f which maintains the database volume 81f. As suggested above, this switch implementation also may be employed in conjunction with the embodiment of Figure 2 and/or Figure 3.

[0066] Usually in implementations such as this there will be a virtualization engine which allows the application system and the database system to treat the different storage systems as a logical single larger storage system. If this circumstance is not present, then all processes which access either storage system must maintain information regarding the address of the particular storage system, rather than relying upon the storage area network to do that. In such a circumstance each of the formatting, allocating and accessing steps requires specific addressing to be achieved. The operation and process flows for this fifth embodiment are the same as described above for the first embodiment. Each process flow

either relies upon the virtualization engine, or provides the specific storage system address from which data is sought to be retrieved, or to which data is to be stored.

[0067] Figure 7 illustrates another embodiment of the invention. In this implementation, a database hub system 150g is added to the first embodiment (Figure 2). The database hub system provides logical access paths to several external (whether existing or legacy) database systems such as system 40g and 40h. It also enables the application system to seamlessly access the different data sources. In current implementations database hub systems do not achieve high performance because they gather data from many sources through relatively low bandwidth LANs. In this embodiment, the database hub system 150g is able to query extensive data from many different sources at high speeds. In Figure 7 the database hub system illustrated can be a well-known system such as the ISO Standard Sequential Query Language/Management of External System (SQLMED). Of course, the database hub implementation also may be employed in conjunction with the embodiment of Figure 2 and/or Figure 3.

[0068] In Figure 7 the overall system architecture is similar to that of the first embodiment, but it includes additional components. The application system includes A-Query requestor 20g. A-Query means a query by the application. This is generated by the application program 10g and sent to the database hub 150g. The hub system includes a process for receiving the query 151g.

[0069] The database hub system 150g receives the A-Query and decomposes it to original E-Queries, meaning queries suitable for application to external databases. The particular decomposition technique will depend upon the particular databases involved. The process 153g sends the queries to the external databases, where they are received by process 50g in database system 40g. They are then executed by process 60g and a return path selected by process 61g. The process of returning the results 70g means that the database result data from the external databases is returned. These external database systems 40g store the result data to the shared volume 90g, and may include the result data gathered from several external databases. In addition, the database system returns the address of the query result data with the key 91g to the database hub system 150g through the LAN. As explained above, an appropriate flag 92g is set.

[0070] The E-Result receiver 154g receives the address of the query result data from each external database that the database hub system 150g requested. This key is checked and the flag set as described above. The E-result composer process 155g causes the database hub system 150g to access each query result data from each external database system through the

storage area network. Then the database hub system 150g decomposes all query result data as the original A-Query originally requested. The process for returning the A-resolved data 156g provides composed query result data from all external query result data. The hub system stores the composed query result data to the shared volume and returns the address of that data with the key 91h to the application system. The flag 92h is also set as explained above. Finally, the A-result receiver 30g receives the address for the results data file 93h and accesses that file. The key is checked and the key is set as explained above.

[0071] As explained above, each of the embodiments of this invention provides unique advantages in comparison to prior art apparatus and systems for handling large amounts of query data or result data. Although preferred embodiments of the invention have been described above, the scope of the invention is defined by the following claims.